Dynamic matching and exogenous thickness

Matthieu Ranger

Abstract. *I present a new software package to simulate dynamic matching markets. The package is open source and distributed under the Quant-Econ project. I discuss exogenous thickness in dynamic matching markets, and why previous literature in dynamic matching diverges in qualitative results. My proposed solution is exploratory simulation studies. I use the package to empirically test the performance of popular matching algorithms in the house allocation problem and in a theoretical model previously studied formally. I find that timing decisions matter more than choice of matching algorithm when the goal is to maximize welfare in house allocation.*

1. Introduction

In many fields, the problem of matching has been studied extensively in various flavors in the single-period (static) setting. In this setting, a set of agents[1], each with individual preferences over outcomes, is considered. The planner then implements a matching using a specified algorithm, observes some properties of the outcome and the problem ends here. The results of this literature have been applied in many settings, including matching organ transplant patients to donors (Roth, Sönmez & Unver, 2004), students to schools (Abdulkadiroğlu & Sönmez, 2003), and residencies for medical school students (Roth, 2002).

This problem is related to the problem of matching in graph theory. Formally, given a graph ***G = (V, E)***, a *matching* is a subgraph of ***G*** where every vertex has degree 1. In other words, the problem consists of finding a set of edges which don't share vertices. We can often represent a matching problem from economics in graph form, where vertices are agents, and an edge represents a potentially feasible match between agents it connects.

When compared to static matching problems, the problem of multi-period (dynamic) matching has been studied sparsely, especially in the formal theoretical setting. Moreover, research on the topic is largely recent. While the dynamic nature of the organ transplant waiting list has been known since Zenios (1999), formal dynamic matching in the general setting has first been studied by Akbarpour, Li and Oveis Gharan (2014).

It's reasonable to assume that the lack of formal theory around dynamic matching problems is partly due to the mathematical difficulty of proving results in the dynamic setting, and not because of lack of interest or applicability. Dynamic processes are notoriously hard to study

[1] Note here that we can consider objects as a special type of agent without any preference on, or utility from, outcomes

formally; even simple processes like the Collatz conjecture remain unprovable with current methods.

It's easy to see that dynamic matching problems provide additional challenges over static matching. Because a chosen matching changes market thickness in future periods, it is important to consider the trade-off between matching agents quickly, and maintaining a market with many potential matchings (a "thick" market). Here, a decision rule on the *time* as well as the *set of agents matched* at said time are important factors over the matching algorithm, as they change market outcomes through the intermediary effect on market thickness.

An example of dynamic matching gone wrong in real economies has been the congestion and subsequent *unravelling* of some professional labor markets (Niederle, Roth & Sönmez, 2008). Because some job markets have many workers and jobs arriving at the same time (for instance, graduation from law or medical school), employers may be incentivized to gain time by making offers early. When unravelling, the appointment dates come earlier and earlier in advance of actual employment. For example, medical residencies, the usual first job of medical school graduates, were to be appointed at a student's fourth year in medical school in 1900, then to half a year before graduation in 1930's, and eventually moved to as much as 2 years before graduation in the 1940's before intervention. Unravelling can cause market failure, specifically when contracts are made before critical information is available (Niederle, Roth & Unver, 2013).

The literature on dynamic matching tends to focus itself piecemeal on specific problems. Depending on the problem at hand, the prescription for approaching the dynamic problem seems to differ. Akbarpour, Li and Oveis Gharan (2014) study a one-sided market (a simplified organ exchange) and find that the benefit of waiting to thicken a market can be tremendous, depending on the discount rate and information structure. Anderson, Ashlagi, Gamarnik, and Kanoria (2015) find in a one-sided item exchange that a policy that maximizes immediate exchanges instead of withholding in for market thickness performs nearly optimally. Baccara, Lee & Yaviv (2015) study a two-sided market and find that an optimal mechanism matches congruent pairs immediately, but keeps a stock incongruent pairs up to a threshold. Anderson (2015) studies an analytic model of the kidney exchange market and finds that the policy of maximizing the number of transplants performed each day is as good as any periodic policy.

Exogenous market thickness in dynamic matching

One reason why the literature differs in conclusions is that the exogenous market conditions differ between models of matching markets. One model, by making subtle assumptions of how a dynamic market evolves over time, may generate qualitatively different optimal conditions than another. These differences are mainly in the processes generating the incoming agents, which, a priori, can be any statistical distribution (degenerate or not).

It is easy to see that a dynamic market's long run thickness is endogenous on the specified matching algorithm and match timing decision rules. To seriously study the performance of various matching methods, then, it's important to have some sort of measure of thickness dictated by the exogenous market conditions.

The exogenous market conditions considered here are the distribution of arrivals, the distribution of agent lifetime if unmatched, and the function dictating match compatibility between agents.

It's unlikely that such a measure would be unifying. First, there is no unifying measure of sparsity in the literature (Hurley & Rickard, 2009). Second, the dynamic graph we are interested in is generated by several stochastic processes which can be distributed arbitrarily.

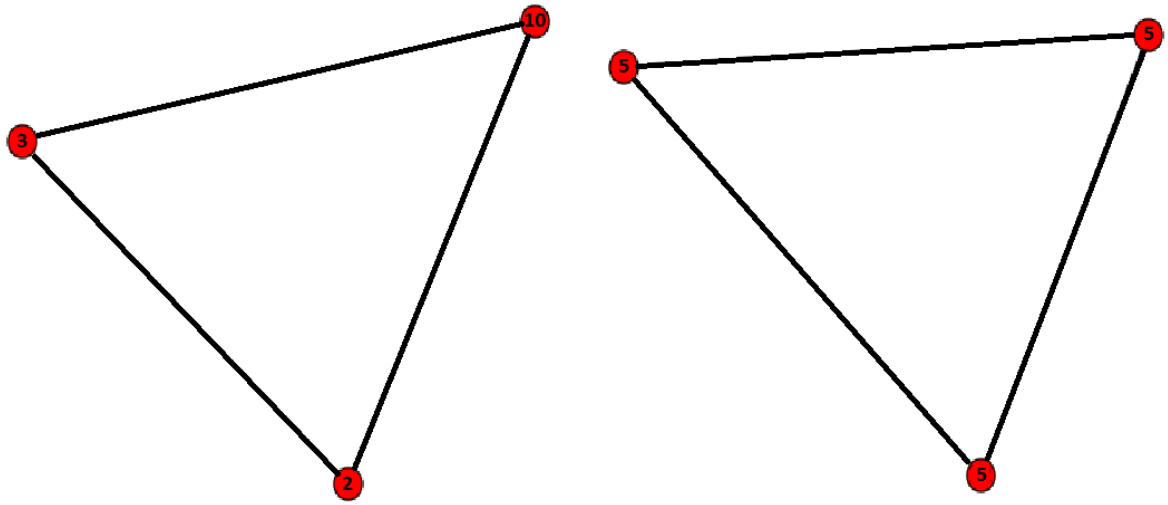

Figure 1. Two graphs. Node labels are lifetime remaining

It's also not clear if the distribution of agent lifetimes should be considered. Take for example the two graphs in figure 1. A measure like the Gini coefficient[2] (Goswami, Murthi & Das, 2016) would consider one much sparser than the other, but other measures, like network density (defined as $\frac{|E|}{|V|(|V|-1)}$) or average vertex degree would consider them equal.

There are few measures of dynamic graphs in the literature. Sarma et al. (2012) define the *dynamic diameter* of a graph, which is the maximum number of rounds it takes for a message to traverse a dynamic network. Kuhn et al. (2010) define *T-interval connectivity*, a stability property where every T consecutive rounds there exists a stable connected spanning subgraph. Doval (2016) proposes a notion for dynamic stable matchings in applications like ours.

To my knowledge, this is the first time the question of sparsity in dynamic graphs is considered.

One intuitive measure would be simply ***dml***, that is the expected degree of an incoming node ***d*** multiplied by the expected arrival rate ***m*** and the expected lifetime ***l***. The problem with such a measure is that it throws out all information about the distribution generation ***d***, ***m***, and ***l***.

For example, it may be more enticing for a planner to wait in a market where the distributions have higher variance. This would be similar to the phenomenon where a worker's reserve wage

[2] This is slight abuse of notation. We would need to weigh the edges by their expected lifetime, and not the nodes, to fit the cited definition.

can increase with higher variance in the market wage distribution in the McCall job search model (Rogerson, Shimer & Wright, 2005).

One formal way would be to consider an ordering by stochastic dominance on the joint distribution. Namely, if the joint distribution $\boldsymbol{dml_1}$ stochastically dominates $\boldsymbol{dml_2}$ in the first order, we can say that the fist is thicker than the second. The problem with this is that most processes are now incomparable, instead of comparable in a flawed way.

Lacking a proper measure, the fallback I propose is to use simulation studies as an exploratory tool.

This paper provides a new way to study general dynamic matching markets in simulations. In **Section 2**, I present the new software package. In **section 3** I test the performance of several algorithms in the dynamic house allocation problem in a simulation study, and then empirically test the findings in Akbarpour, Li and Oveis Gharan (2014) and extend these findings to different algorithms and timing decision rules.

2. The *MatchingMarkets.py* package

MatchingMarkets.py is a python[3] programming language package distributed open-source under the Quant-Econ project (Sargent & Starchuski, 2013). It is built to model and study arbitrary dynamic matching markets. The package is based around the concepts of agents, generators, a market, and matching algorithms.

The **agents** in the market represent all relevant members of a matching market. They are differentiated through a *type,* an attribute they hold which can be any python object. For instance, we can represent males with a type of `0` and females with a type of `1` in a stable marriage problem. We could also represent the blood types in a kidney exchange with character strings, like for example "**AB+**" or "**O-**". Agents also maintain maps for the utility of matching with each other agent, and the probability of success of a match with each other agents (which is simply `0` or `1` in the case of deterministic matches). Agents can be further differentiated by a name, and a discount rate on the utility of a match.

Generators are functions specified by the user which dictate the arrival of new agents in the market at each period[4]. This includes the *arrival rate* (usually a Poisson distribution draw, but may be arbitrarily changed), the lifetime of agents (which can be stochastic and heterogenous), a function which generates the type of agents (to create the structure of the market) and the function which defines the structure of the agent's compatibility and utility maps.

[3] See python.org

[4] The package only supports discrete time simulation. Continuous time problems can be approximated through many time periods and constraints on generator functions

The **market** is simply a container for all agents. It maintains a graph of matches and is responsible for tracking interactions and updating periods. The market graph can be interactively visualized as seen in figure 2.

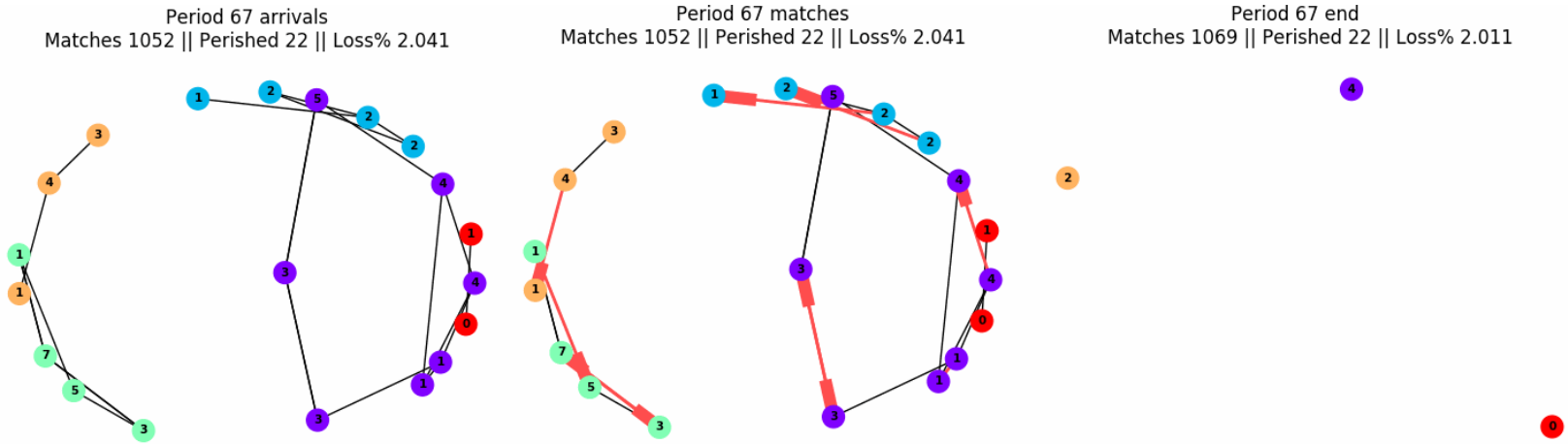

Figure 2. *A visualization of one period in a dynamic market. Here, nodes are agents, the node color represents their type, edges signify match compatibility. Bold edges in the graph are matches implemented. A node's label is his remaining lifetime if unmatched.*

The **matching algorithm** is defined in two parts. First, the match timing decision rule is decided by a matching *meta-algorithm,* which specifies which agents are to be attempted to match this period. Then, this information is passed into a matching algorithm typical of a static matching problem. The agents passed can attempt matches with the rest of the market, or only between themselves.

A growing library of predefined generator functions and matching algorithms is being implemented with a default installation of the package. Tutorials and documentation can be found in the projects open source repository[5] through the Quant-Econ project.

3.1 Empirical results of dynamic house allocation

The *house allocation problem* is a one-sided matching problem, where there are a finite and equal number of agents and indivisible goods. An allocation is an assignment of houses to agents, such that each agent receives exactly one house. Note here that an agent can "match himself" if he so desires. This characteristic of the problem makes a house allocation problem inherently thicker, because it implies the graph is complete always.

We look at different matching time intervals on several algorithms for this problem. The algorithms considered are **Serial Dictatorship** and its randomized variant, the **TTC** algorithm (Shapley and Scarf 1974). We also test the graph-theoretic **maximum weight matching** algorithm, based on Jack Edmonds' "blossom" method for directed graphs (Galil, 1986) which finds a graph matching that maximizes the edge weights (utility in this case). Lastly, I test the

[5] https://github.com/QuantEcon/MatchingMarkets.py

maximum cardinality matching, a greedy algorithm matching a maximal number of edges in a graph.

Note that while the Serial Dictatorship and TTC algorithms have several desirable properties proven. Both are strategyproof and Pareto efficient, but do not minimize justified envy (Abdulkadiroğlu & al. 2017). Graph theoretic matching algorithms, on the other hand, have no such guarantees – they are as dictatorial as it gets.

In this simulation, we observe a market where agents "trickle in" at a slower pace of 0.3 per period and live on average 35 periods (those are the λ in Poisson draws). The utility of matches between agents heterogenous and uniformly randomly distributed in **[0,1]**. We observe the total utility of the market for agents who don't discount, and agents with a discount rate of 0.95.

Table 2. Welfare in dynamic house allocation problem

5500 periods, average lifetime of 35 periods

$\delta = 1$

Match Frequency	Serial Dictatorship	Random Serial Dictatorship	TTC	Maximal Weight Matching	Maximal Cardinality Matching
Every Period	869.1 (38.8)	858.9 (19.3)	872.5 (18.7)	830.4 (18.1)	832.5 (21.1)
3 periods	904.6 (26.2)	915.7 (21.3)	929.3 (38.4)	818.0 (23.9)	833.9 (30.2)
10 periods	1012.3 (34.1)	1025.1 (27.0)	1015.6 (33.4)	818.0 (27.7)	796.6 (20.6)
35 periods	1079.3 (23.8)	1082.35 (26.9)	956.4 (28.2)	736.6 (34.5)	554.8 (14.5)

$\delta = 0.95$

Match Frequency	Serial Dictatorship	Random Serial Dictatorship	TTC	Maximal Weight Matching	Maximal Cardinality Matching
Every Period	860.4 (22.1)	864.9 (18.2)	852.0 (23.7)	773.8 (30.6)	825.2 (26.3)
3 periods	876.3 (30.1)	869.1 (26.5)	873.3 (28.4)	742.6 (27.1)	771.4 (22.4)
10 periods	813.6 (25.0)	815.6 (25.8)	804.2 (22.2)	623.7 (18.4)	582.5 (12.6)
35 periods	488.0 (19.0)	534.4 (13.3)	470.3 (15.2)	362.1 (11.5)	256.8 (13.5)

*Standard deviations in parentheses. Agents' utility from matching is drawn heterogeneously for each pair in uniform **[0,1]** at market arrival. Welfare is the sum of utility of matched agents.*

We can see from the results of the simulations[6] in table 2 that, the discount rate must be considered when choosing the timing rule. Moreover, the difference between timing rules is greater than the difference between the popular algorithms. Unsurprisingly, graph theoretic algorithms tend to perform worse in house allocation than algorithms designed and studied for this problem.

3.2 Empirical Results in Akbarpour & al (2014)

We now shift attention to Akbarpour, Li and Oveis Gharan (2014) (ALG). ALG studies a continuous time model where agents arrive according to a Poisson process. The agents' lifetime is also determined according to a Poisson process at arrival; if an agent does not find a match in his lifetime in the market, he is said to *perish.* Matches are *bilateral*, that is, if agent **α** matches with agent **β**, then **β** is automatically matched with **α**. An agent receives utility of 1 if matched, discounted at a rate **δ**. If an agent perishes, he has no utility. Each pair of agent regards a transaction as acceptable with some probability uniform to the market.

Agents are matched arbitrarily under various decision rules. That is, if an agent is decided to match at a period by the decision rule, he will be matched with an arbitrary agent in the rest of the market. The decision rules studied in ALG are the **Greedy** decision rule, where agents are attempted to match immediately upon arrival, the **Patient** decision rule where agents are attempted to match at their last period of life in the market before perishing, and a variation on the Patient algorithm, where agents may be attempted a match earlier. ALG compare the performance of these algorithms on *Loss* (ratio of perished agents in the market) and *Welfare* (sum of utility of agents in the market).

These decision rules are then compared to the theoretical optimal rule and a rule based on omniscience, which has knowledge of all future agents when making its decisions. Because these markets are exponential in size, we cannot use the tools from dynamic programming to solve for the optimum matching algorithm, even for small markets.

ALG find that waiting to thicken the market can yield substantial welfare gains, and the result robust to the presence of waiting costs. They also find that the naive waiting algorithm can yield close to optimal results.

Using the *MatchingMarkets.py* package, we can simulate the ALG results in discrete time. In **Table 1**, we see Monte Carlo simulation results for ALG's model under various algorithms and timing decision rules. Because the model matches bilaterally, and ALG's original timing rules attempted matches on individual agents, we cannot compare the performance of this model with popular algorithms that match pools of agents like TTC or deferred acceptance.

The timing rules (meta-algorithms) considered are **Greedy** and **Patient**, as previously defined, the **Always Match** rule, where we attempt all agents every period, and **Periodic(a)**, where we attempt to match all agents in the market every **α** periods.

[6] The code for all simulations is freely available where the package is distributed

Table 1. Loss% *Simulation results of ALG (2014) model*

	Meta-Algorithm	**Arbitrary Match**	**Serial Dictatorship** ordered by remaining lifetime	**Random Serial Dictatorship**
Thin Market **d** = 0.1 **m** = 0.7 **lifetime** = 8	*Greedy*	0.5395 (0.0067)	0.5427 (0.0080)	0.5418 (0.0096)
	Always Match	0.0053 (0.0006)	0.0042 (0.0001)	0.0057 (0.0022)
	Patient	0.0169 (0.0018)	0.0196 (0.0010)	0.0164 (0.0025)
	Periodic (2)	0.0062 (0.0014)	0.0047 (0.0012)	0.0065 (0.0018)
	Periodic (3)	0.0142 (0.0027)	0.0083 (0.0021)	0.0142 (0.0026)
	Periodic (4)	0.0264 (0.0009)	0.0139 (0.0019)	0.0264 (0.0028)
Thick Market **d** = 0.3 **m** = 3 **lifetime** = 8	*Greedy*	0.1927 (0.0012)	0.1921 (0.0025)	0.1943 (0.0031)
	Always Match	0.0001 (0.000)	0.0000 (0.0001)	0.000 (0.0001)
	Patient	0.0006 (0.0002)	0.0008 (0.0004)	0.0005 (0.0001)
	Periodic (2)	0.0006 (0.0001)	0.0002 (0.0001)	0.0009 (0.0004)
	Periodic (3)	0.0026 (0.0006)	0.0013 (0.0003)	0.0031 (0.0005)
	Periodic (4)	0.0085 (0.0005)	0.0046 (0.0006)	0.0074 (0.0006)

***m** is the arrival rate of agents, **Lifetime** is the average number of periods before perishing. Both are λ parameter in a Poisson distribution. **d** is the average probability two agents are compatible. **Loss%** is the ratio of perished agents to total agents in the market. Standard deviation in parentheses. These figures are based on 5 Monte Carlo runs of 3500 periods per simulation.*

The algorithms used in table 1 are **Arbitrary Match**, where an agent is matched with a random compatible agent, the classic **Serial Dictatorship**, where agents to be matched in a period are ordered by their remaining lifetime and then sequentially match their preferred agent, and **Random Serial Dictatorship**, which is a variant of serial dictatorship where the order is random.

It's important to note that, since the utility of a match is 1 discounted by time in ALG (2014) regardless of choice, serial dictatorship mainly differs by the ordering of matched agents. Also note the poor performance of the Greedy algorithm – since agents are only attempted to match

upon arrival or if a compatible agent matches them, the overall performance of this timing rule is poor, even for thick markets.

We can clearly see from the results that the timing decision rule matters tremendously in outcomes in ALG (2014)'s model, and tends to matter more than the matching algorithm chosen.

Conclusion

I present a new software package to simulate dynamic matching markets. The package is open source and distributed under the Quant-Econ project.

I discuss exogenous thickness in dynamic matching markets, and why the previous literature on dynamic matching differs in results. I use the package to empirically test the performance of popular matching algorithms in the house allocation problem and in a theoretical model previously studied formally. I find that timing decisions matter more than choice of matching algorithm when the goal is to maximize welfare in house allocation.

References

ABDULKADIROGLU, Atila and Tayfun SONMEZ. (2003). "School Choice: A Mechanism Design Approach," *American Economic Review, 93(3): 729-747.*

ABDULKADIROGLU, A. CHE, Y-K., PATHAK, P. ROTH, A. E. and TERCIEUX, O. (2017) "Minimizing justified envy in school choice," *NBER Working paper*

AKBARPOUR, M., S. LI, and S. OVEIS GHARAN (2014). "Dynamic matching market design," *mimeo.*

ANDERSON, Ross, Itai ASHLAGI, David GARMANIK, and Yash KANORIA. 2015. "Efficient Dynamic Barter Exchange," *mimeo.*

ANDERSON, Ross. (2009). "Stochastic models and data driven simulations for healthcare operations," (Unpublished doctoral thesis) Cornell University, U.S.

BACCARA, Mariagiovanna and LEE, SangMok and YARIV, Leeat, (2015). "Optimal Dynamic Matching ," *mimeo.* http://dx.doi.org/10.2139/ssrn.2641670

DOVAL, Laura. (2009). "A Theory of Stability in Dynamic Matching Markets," *mimeo*

GALIL, Zvi. (1986). "Efficient Algorithms for Finding Maximum Matching in Graphs," *ACM Computing Surveys*.

GOSWAMI, S. MURTHY, A. and DAS, A. (2016) "Sparsity measure of network graph: Gini index," *mimeo*

HURLEY, N. and S. RICKARD. (2009). "Comparing measures of sparsity." *IEEE Trans. Inf. Theory*, pages 4723–4741

KUHN, F., LYNCH, N., OSHMAN, O. (2010) "Distributed computing in dynamic networks," *Proceedings of the forty-second ACM symposium on Theory of computing* Pages 513-522

NIEDERLE, M., ROTH, A. E. and SONMEZ, T. (2008). "Matching and Market Design," *New Palgrave Dictionary of Economics*, second edition.

NIEDERLE, M., ROTH, A. E. and UNVER, U. (2013). “Unraveling results from comparable demand and supply: An experimental investigation,” *Games*, 4:2, 243-282

ROGERSON, Richard, Robert SHIMER and Randall WRIGHT. 2005. "Search-Theoretic Models of the Labor Market: A Survey." Journal of Economic Literature, 43(4): 959-988.

ROTH, A. E., T. SONMEZ, and U. Unver (2004). “Kidney Exchange," *Quarterly Journal of Economics*, 119, 457-488.

ROTH, A. E. (2002). The Economist as Engineer: Game Theory, Experimentation, and Computation as Tools for Design Economics," *Econometrica*, 70, 1341-1378.

SARGENT, T., & STARCHUSKI, J. (2013). Quantitative economics. *URL http://quant-econ. Net*.

SARMA, Das. A., LALL, A., NANONGKAI, D., TREHAN, A. (2012) “Dense Subgraphs on Dynamic Networks.” *Distributed Computing*. vol 7611.

SHAPLEY, Lloyd and Herbert SCARF (1974). “On Cores and Indivisibility," *Journal of Mathematical Economics*, 1, 23-28.

ZENIOS, Stefanos A. (1999). “Modeling the transplant waiting list: A queueing model with reneging,” *Queueing Systems*, 31, 239-251.